Voting Management System

Database Systems - CS F212

Spring Semester Project 2023

ABIR ABHYANKAR

2021A7PS0523P

Materials -

https://drive.google.com/drive/folders/1d7idf7iq8Oz2rLDgYPaFb_-yMRBKdhnM

Documentation Preface

This application is a relational database voting management system designed to manage elections efficiently and securely. It is a comprehensive system that sets up elections and adds candidates, manages voters, candidates, and admins smoothly.

Installation Instructions

To use this Application, follow these instructions:

● Download and install MySQL Community Server:

1. Go to https://dev.mysql.com/downloads/mysql/ to download the MySQL Community Server. Choose the Appropriate Version

● Configure MySQL:

1. Once installed, you need to configure MySQL to run on your machine.
2. During installation, you will be prompted to set a root password for MySQL. Make sure to choose a strong password and keep it secure.
3. After installation, you can access the MySQL Server from the command line using the command "mysql -u root -p" and enter the root password when prompted.

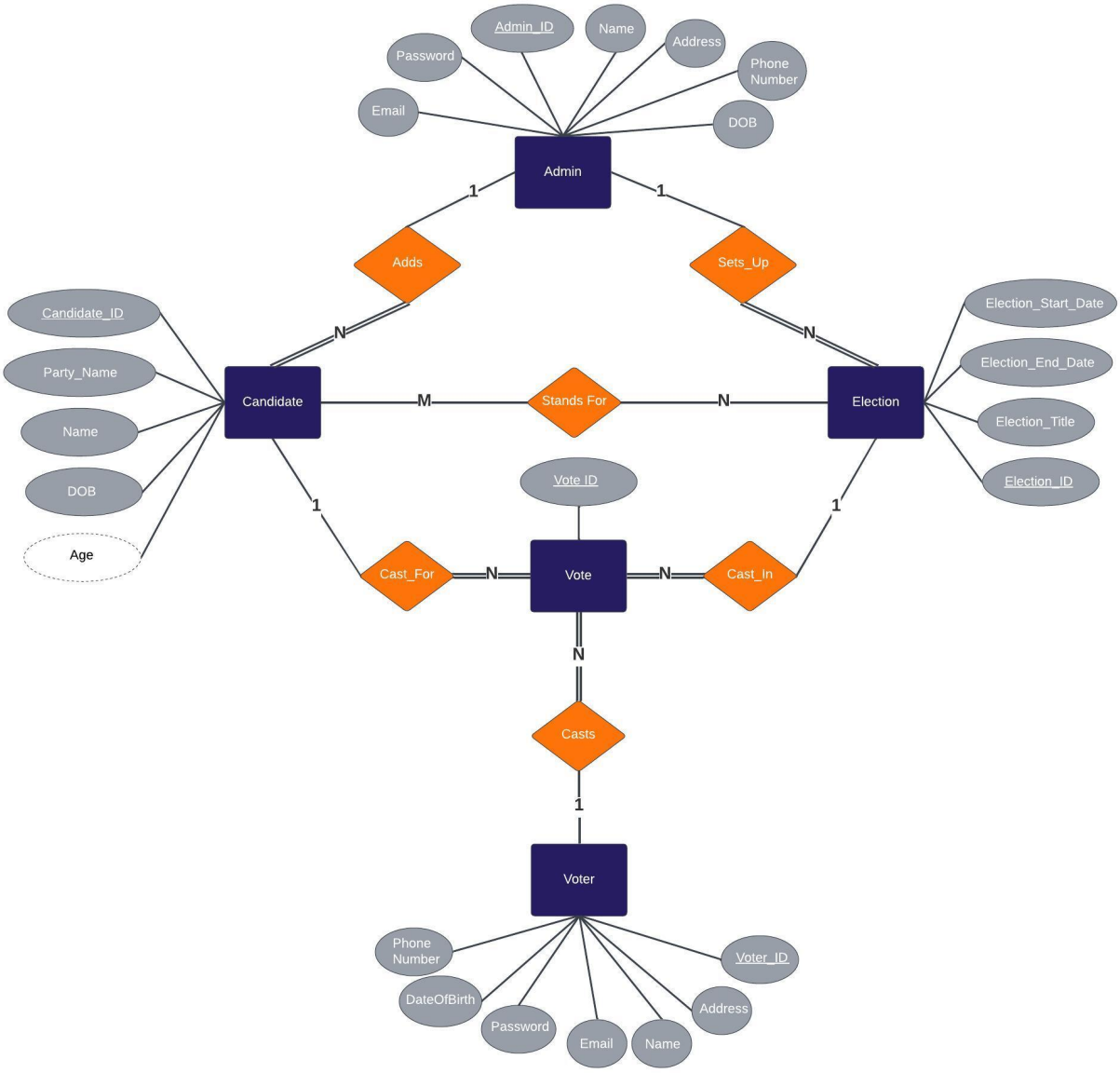
● Run the Application:

1. Run the Proj13_runFirst.sql File to Create the necessary Tables and populate them with realistic data in your PC, do install the required packages mentioned on the last page.
   Commands - pip install tkinter, pip install mysql, Installation of Python is also Essential.
2. You need to access the Python file named "Queries.py", and edit the user ID and password of your MySQL Installation on Lines **9 & 10**. On entering this information, run the "User_Mode_Page.py" File to access the application.

The Application is capable of running many different types of Insertion, Updation, Deletion and Retrieval of Data. Combined with a Very User-Friendly GUI, you can use all features to the maximum extent possible.

Voting Management System

ER DIAGRAM



Entities:

1. **Admin** - This entity represents the Administrator of the Voting Management System, who is responsible for setting up the election, adding candidates, and ensuring overall system security. The attributes of this entity are:

- **Admin_ID** (Primary Key): A Unique ID to Identify Each Administrator.
- **Password**: The Password of the Administrator, which is used to Log In to the system.
- **Name**: The Name of the Administrator.
- **Email**: The Email Address of the Administrator.
- **Address**: The Home Address of the Administrator.
- **Phone_Number**: The Phone Number of the Administrator.
- **DOB**: The Date of Birth of the Administrator.

2. **Candidate** - This entity represents a Candidate who stands up and contests Elections for a Party/Group. The attributes of this entity are:

- **Candidate_ID** (Primary Key): A Unique ID to Identify Each Candidate.
- **Name**: The Name of the Candidate.
- **DOB**: The Date of Birth of the Candidate.
- **Age** (Derived Attribute): The Age of the Candidate.
- **Party_Name**: The Party to which the Candidate belongs.

3. **Election** - This entity represents an event/entity in which Candidates contest and Voters Vote. The attributes of this entity are:

- **Election_ID** (Primary Key): A Unique ID to Identify Each Election.
- **Election_Start_Date**: The First Date on which Voters can Vote in the given Election.
- **Election_End_Date**: The Last Date on which Voters can Vote in the given Election.
- **Election_Title**: The Title of the given Election.

4. **Voter** - This entity represents the people/group who can Vote in Elections and elect their preferred Candidate. The attributes of this entity are:

- **Voter_ID** (Primary Key): A Unique ID to Identify Each Voter.
- **Password**: The Password of the Voter, which is used to Log In to the system and Cast Votes.
- **Name**: The Name of the Voter.
- **Email**: The Email Address of the Voter.

- **Address**: The Home Address of the Voter.
- **Phone_Number**: The Phone Number of the Voter.
- **DateOfBirth**: The Date of Birth of the Voter.

5. **Vote** - This entity represents the will and power of a Voter to Elect a Candidate of the Voter's Choice in a Given Election. The attributes of this entity are:

- **Vote_ID** (Primary Key): A Unique ID to Identify Each Vote.

Relationships:

1. **Adds** - This Relationship connects the **Admin** Entity to the **Candidate** Entity. It is denoting the fact that each **Admin** can add multiple **Candidates**. The Cardinality of this Relationship is **1 : N**, meaning that **Each Admin** can add **Many Candidates**, but **Each Candidate** has been added by a **Single Admin**.

2. **Sets_Up** - This Relationship connects the **Admin** Entity to the **Election** Entity. It is denoting the fact that each **Admin** can set up multiple **Elections**. The Cardinality of this Relationship is **1 : N**, meaning that **Each Admin** can set up **Many Elections**, but **Each Election** has been set up by a **Single Admin**.

3. **Stands_For** - This Relationship connects the **Candidate** Entity to the **Election** Entity. It is denoting the fact that each **Candidate** can stand for one or more **Elections**, and each **Election** can have one or more **Candidates** contesting. The cardinality of this relationship is **M : N**, meaning that **Candidates** can stand for **Multiple Elections** and **Elections** can have **Multiple Candidates** contesting.

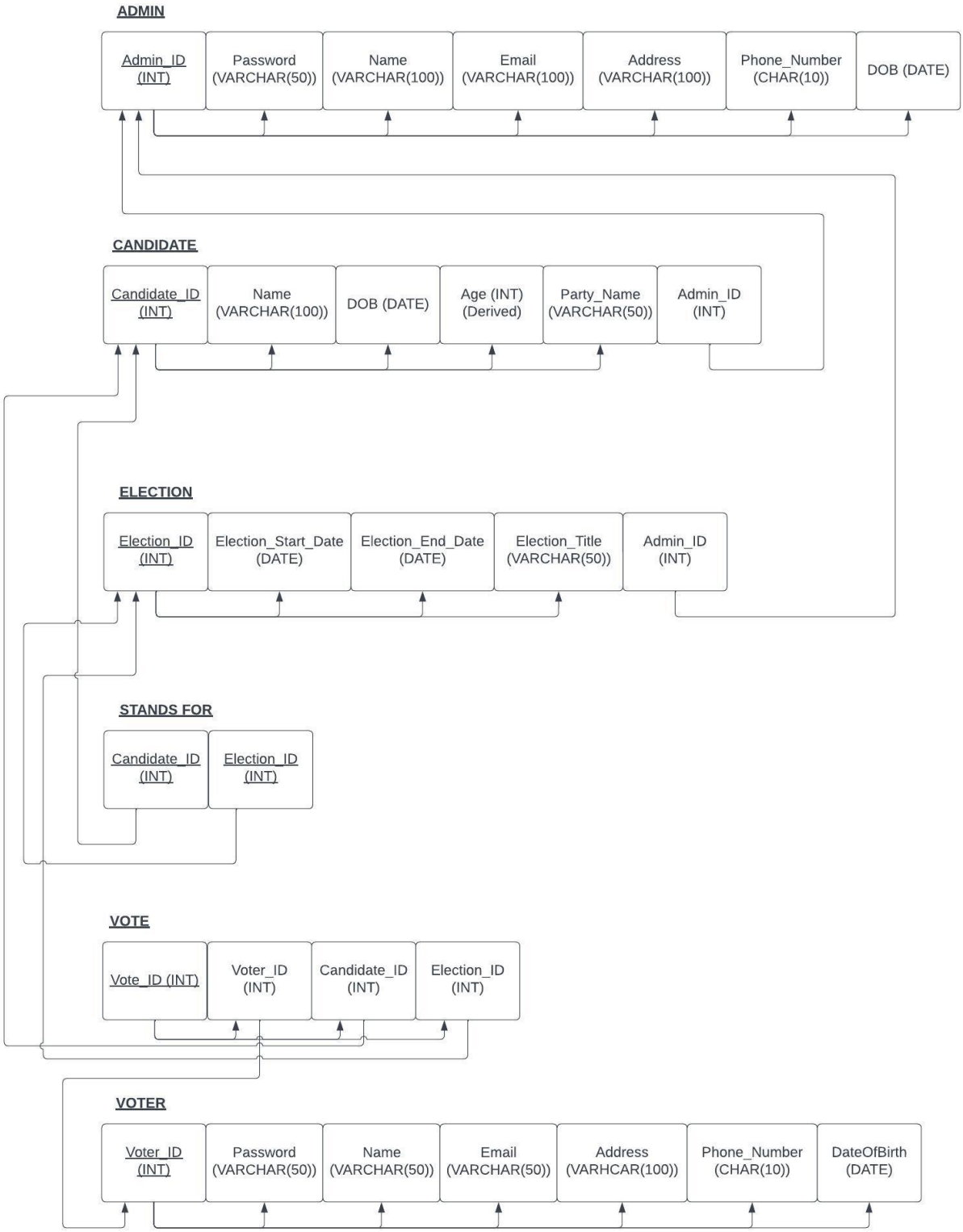
4. **Cast_For** - This Relationship connects the **Candidate** Entity to the **Vote** Entity. It is denoting the fact that multiple **Votes** can be cast for a single **Candidate**. The Cardinality of this Relationship is **1 : N**, meaning that **Each Candidate** can have **Multiple Votes** behind him, but **Each Vote** is cast for a **Single**

Candidate.

5. **Cast_In**- This Relationship connects the **Election** Entity to the **Vote** Entity. It is denoting the fact that multiple **Votes** can be cast in a single **Election**. The Cardinality of this Relationship is **1 : N**, meaning that **Each Election** can have **Multiple Votes** casted in it, but **Each Vote** is cast in a **Single Election**.

6. **Casts** - This Relationship connects the **Vote** Entity to the **Vote** Entity. It is denoting the fact that a single **Voter** can cast multiple votes in different elections. The Cardinality of this Relationship is **1 : N**, meaning that **Each Voter** can cast **Multiple Votes**, but **Each Vote** is cast by a **Single Voter**.

Relational Model (Before Normalization)



1. **Admin Table** - This entity represents the Administrator of the Voting Management System, who is responsible for setting up the election, adding candidates, and ensuring overall system security. The attributes of this entity are:

- **Admin_ID** (Primary Key): A Unique ID to Identify Each Administrator.
- **Password**: The Password of the Administrator, which is used to Log In to the system.
- **Name**: The Name of the Administrator.
- **Email**: The Email Address of the Administrator.
- **Address**: The Home Address of the Administrator.
- **Phone_Number**: The Phone Number of the Administrator.
- **DOB**: The Date of Birth of the Administrator.

2. **Candidate Table** - This entity represents a Candidate who stands up and contests Elections for a Party/Group. The attributes of this entity are:

- **Candidate_ID** (Primary Key): A Unique ID to Identify Each Candidate.
- **Name**: The Name of the Candidate.
- **DOB**: The Date of Birth of the Candidate.
- **Age** (Derived Attribute): The Age of the Candidate.
- **Party_Name**: The Party to which the Candidate belongs.
- **Admin_ID** (Foreign Key): A Unique ID to Identify Each Administrator. Since the '**Sets_Up**' relationship between **Admin** and **Candidate** was **1 : N** while converting the ER diagram to a Relational Schema this relationship was removed and **Admin_ID** which is the Primary Key of the **Admin Table** was added as a Foreign Key to the **Candidate Table**.

3. **Election** - This entity represents an event/entity in which Candidates contest and Voters Vote. The attributes of this entity are:

- **Election_ID** (Primary Key): A Unique ID to Identify Each Election.
- **Election_Start_Date**: The First Date on which Voters can Vote in the given Election.
- **Election_End_Date**: The Last Date on which Voters can Vote in the given Election.
- **Election_Title**: The Title of the given Election.

- **Admin_ID** (Foreign Key): A Unique ID to Identify Each Administrator. Since the '**Adds**' relationship between **Admin** and **Election** was **1 : N** while converting the ER diagram to a Relational Schema this relationship was removed and **Admin_ID** which is the Primary Key of the **Admin Table** was added as a Foreign Key to the **Election Table**.

4. **Voter** - This entity represents the people/group who can Vote in Elections and elect their preferred Candidate. The attributes of this entity are:

- **Voter_ID** (Primary Key): A Unique ID to Identify Each Voter.
- **Password**: The Password of the Voter, which is used to Log In to the system and Cast Votes.
- **Name**: The Name of the Voter.
- **Email**: The Email Address of the Voter.
- **Address**: The Home Address of the Voter.
- **Phone_Number**: The Phone Number of the Voter.
- **DateOfBirth**: The Date of Birth of the Voter.

5. **Vote** - This entity represents the will and power of a Voter to Elect a Candidate of the Voter's Choice in a Given Election. The attributes of this entity are:

- **Vote_ID** (Primary Key): A Unique ID to Identify Each Vote.
- **Voter_ID** (Foreign Key): A Unique ID to Identify Each Administrator. Since the '**Casts**' relationship between **Voter** and **Vote** was **1 : N** while converting the ER diagram to a Relational Schema this relationship was removed and **Voter_ID** which is the Primary Key of the **Voter Table** was added as a Foreign Key to the **Vote Table**.
- **Candidate_ID** (Foreign Key): A Unique ID to Identify Each Administrator. Since the '**Cast_For**' relationship between **Candidate** and **Vote** was **1 : N** while converting the ER diagram to a Relational Schema this relationship was removed and **Candidate_ID** which is the Primary Key of the **Candidate Table** was added as a Foreign Key to the **Vote Table**.
- **Election_ID** (Foreign Key): A Unique ID to Identify Each Administrator. Since the '**Cast_In**' relationship between **Election** and **Vote** was **1 : N** while converting the ER diagram to a Relational Schema this relationship was removed and **Election_ID** which is the Primary Key of the **Election Table** was added as a Foreign Key to the **Vote Table**.

Conversion to 3NF

To convert the relational model to 3NF, we followed the following steps:

1. First we will convert the Pre-Normalization Relational Schema into 1NF.

   Converting to 1NF :-

   - On looking at the relational schema, we observe that none of the attributes in the relational schema is multivalued or composite.
   - Hence, we can conclude that the relational schema is already in 1NF.

2. Next we will convert the Relational Schema from 1NF to 2NF.

   Converting to 2NF :-

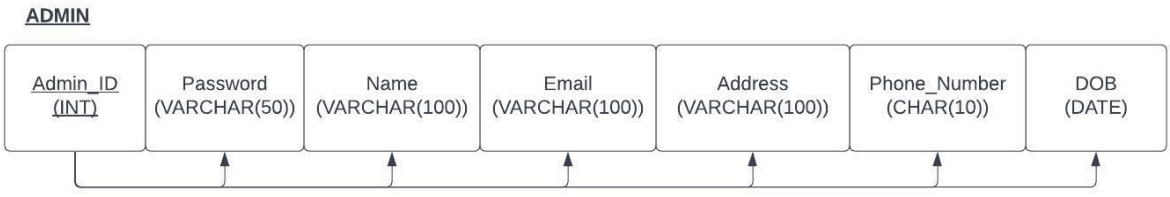
   - On looking at the relational schema, we observe that no sub-part of any candidate key of any table could uniquely identify any other attributes i.e every non-prime attribute is fully functional dependent.
   - Hence, we can conclude that the relational schema is already in 2NF.

3. We observe that our Relational Schema is in 2NF, now we will analyse and try to convert it to 3NF.

Converting to 3NF :-

1. First we have to identify the functional dependencies in each table

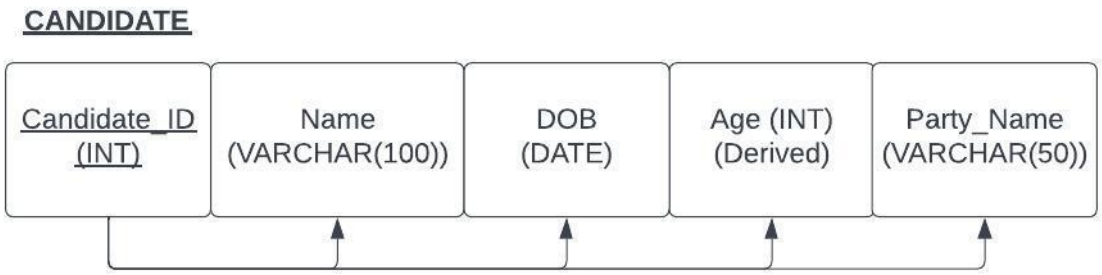
(i) **Admin**



Functional dependency:

- **Admin_ID** → Password, Name, Email, Address, Phone_Number & DOB

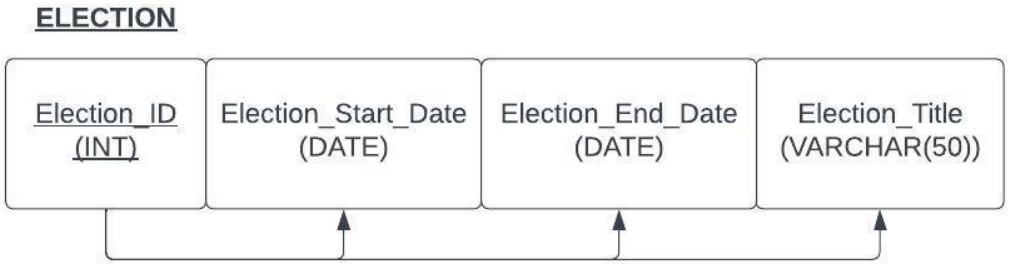
(ii) **Candidate**



Functional dependency:

- **Candidate_ID** → Name, DOB, Age & Party_Name

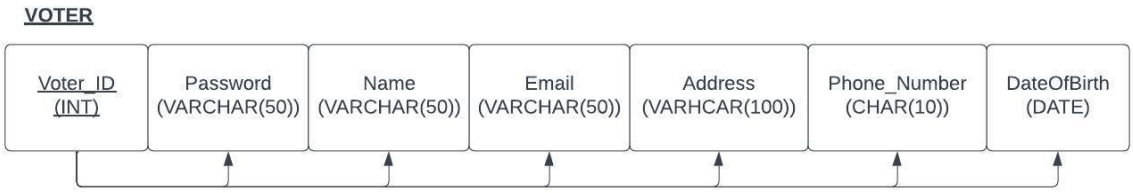
(iii) **Election**



Functional dependency:

- **Election_ID** → ELection_Start_Date, Election_End_Date & Election_Title

(iv) **Voter**



Functional dependency:

- **Voter_ID** → Password, Name, Email, Address, Phone_Number & DateOfBirth

As we can see for the 4 tables above, all attributes of each table are dependent solely on the primary key, and thus they are in 3NF.

Similarly for the rest of the tables (**Vote** and **Stands_For**) all attributes are dependent only on the primary key and thus they are already in 3NF form.

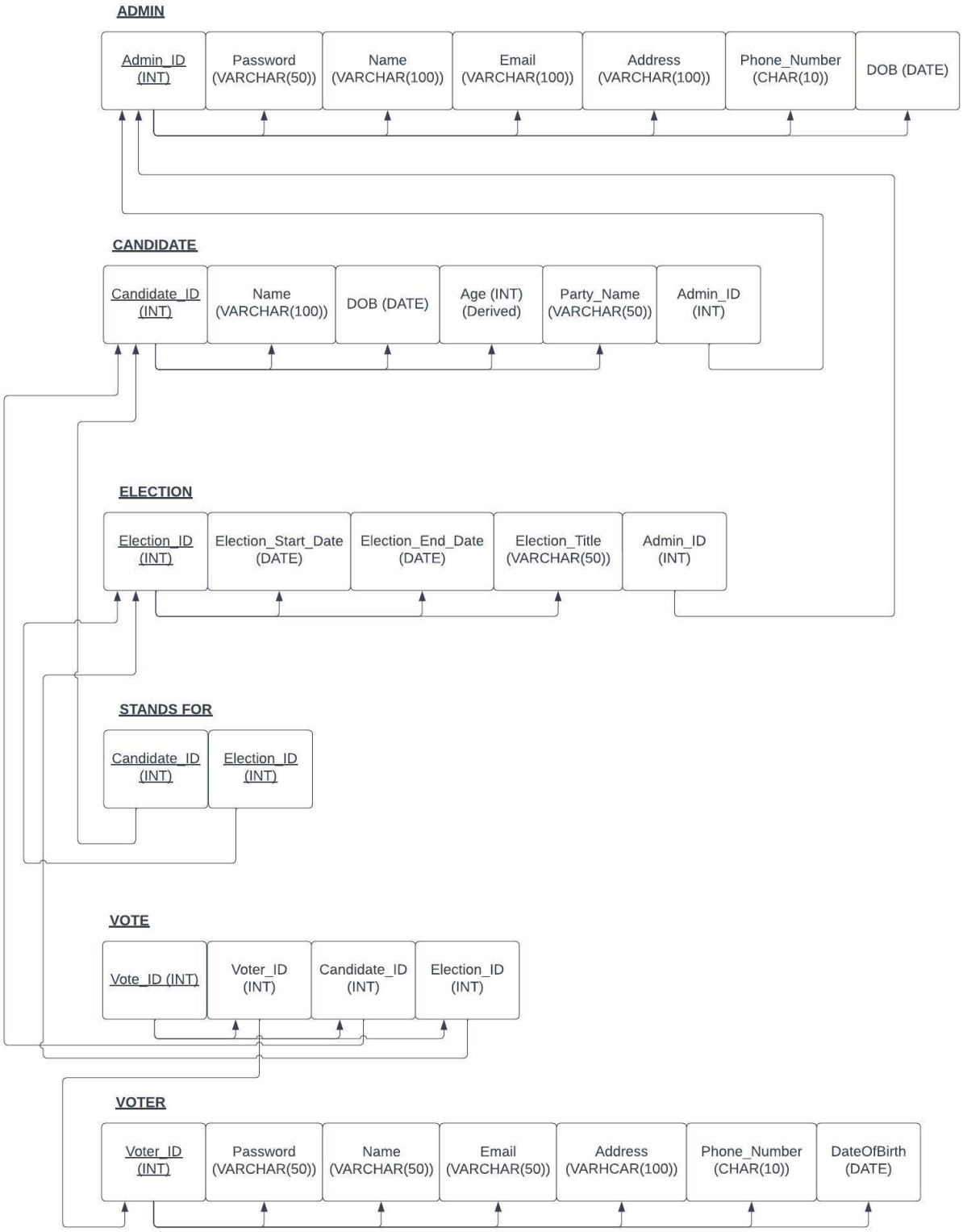
2. Now we will check for transitive dependencies and if we find any we will remove them by creating new tables.

In all tables, all attributes of the table are solely dependent on their respective primary keys. Thus there are no transitive dependencies of any sort, and we can safely say that the Relational Schema is in 3NF.

All the tables are now in 3NF.

The above relational schema has been normalized and is now in 3NF form.

Relational Model (After Normalization)



SQL Queries Explanation

I have inserted Sample Data into All the Queries. You can replace the Sample data with any random data of the appropriate type and run the Queries smoothly. You can also refer to Procedures.sql to gain a better understanding of the Queries.

→ Query to Add Voters

```
-- Query To Add Voter
INSERT INTO Voter(Password,Name,Email,Address,Phone_Number,DateOfBirth)
VALUES('password16','Abir Abhyankar','abirabh20170@gmail.com',
'Salunke Vihar, Delhi - 111001','9103025510','2004-01-26');
```

We can replace the Values with our Own and Run the program efficiently.

This Query is used to Insert a new Voter into the **Voter** Table. It takes the following parameters:

- **Password**: The Password of the Voter (VARCHAR(50)).
- **Name**: The Name of the Voter (VARCHAR(50)).
- **Email**: The Email Address of the Voter (VARCHAR(50)).
- **Address**: The Home Address of the Voter (VARCHAR(100)).
- **Phone_Number**: The Phone Number of the Voter (CHAR(10)).
- **DateOfBirth**: The Date of Birth of the Voter (DATE).

Integration is such to catch all cases of Information in Invalid Formats and Showcase the Errors to the User to fix. Highly Controlled Database Access. Great help in validating Queries with a DATE type parameter.

→ Query to Add Candidates

```
-- Query To Add Candidates
INSERT INTO Candidate(Name,DOB,Age,Party_Name,Admin_ID)
VALUES('Arjun Abhyankar','2000-09-09','22','Liberty League','1');
```

This Query is used to Insert a new Candidate into the **Candidate** Table. It takes the following parameters:

- **Name**: The Name of the Candidate (VARCHAR(100)).
- **DOB**: The Date of Birth of the Candidate (DATE).
- **Age**: The Age of the Candidate (INT).
- **Party_Name**: The Name of the Party of which the Candidate is part of (VARCHAR(50)).
- **Admin_ID**: The **Admin_ID** of the Admin who added Candidate (INT).

Integration is such to catch all cases of Information in Invalid Formats and Showcase the Errors to the User to fix. Highly Controlled Database Access. Great help in validating Queries with a DATE type parameter.

→ Query To Register A Vote

```
-- Query To Register A Vote
INSERT INTO Vote(Voter_ID,Candidate_ID,Election_ID) VALUES(2,5,1);
```

This Query is used to Register A Vote into the **Vote** Table. It takes the following parameters:

- **Voter_ID**: The Voter_ID of the Voter who cast the Vote (INT).
- **Candidate_ID**: The Candidate_ID of the Candidate the Vote is Cast For (INT).
- **Election_ID**: The Election_ID of the Election the Vote is Cast In (INT).

→ Query to Set Up an Election

```
-- Query To Add Elections
INSERT INTO Election(Election_Start_Date,Election_End_Date,Election_Title,Admin_ID)
VALUES('2024-05-03','2024-05-07','Lok Sabha Elections',1);
```

This Query is used to Set Up a New Election into the **Election** Table. It takes the following parameters:

- **Election_Start_Date**: The First Date on which Voters can Vote in the given Election. (DATE)
- **Election_End_Date**: The Last Date on which Voters can Vote in the given Election. (DATE)
- **Election_Title**: The Title of the given Election. (DATE)
- **Admin_ID**: The **Admin_ID** of the Admin who added Candidate (INT).

Integration is such to catch all cases of Information in Invalid Formats and Showcase the Errors to the User to fix. Highly Controlled Database Access. Great help in validating Queries with a DATE type parameter.

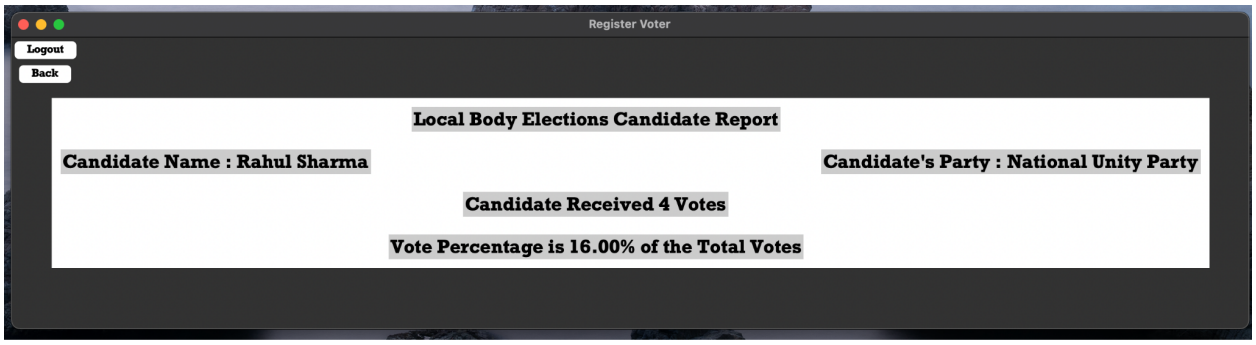
→ Query to Retrieve the Number of Votes For A Candidate in a Given Election

```
-- Query To Count Total Number of Votes Received by a Candidate in an Election
SELECT COUNT(*) FROM VOTE WHERE Candidate_ID = 1 AND Election_ID = 1;
```

This Query is used to Count the Number of Votes For a **Candidate** in a given **Election**. It takes the following parameters:

- **Candidate_ID**: The Candidate_ID of the Candidate the Vote is Cast For (INT).
- **Election_ID**: The Election_ID of the Election the Vote is Cast In (INT).

It gives us the Count of Votes For a Candidate. In this example, the Candidate Rahul Sharma got **4 Votes** in the Local Body Elections.



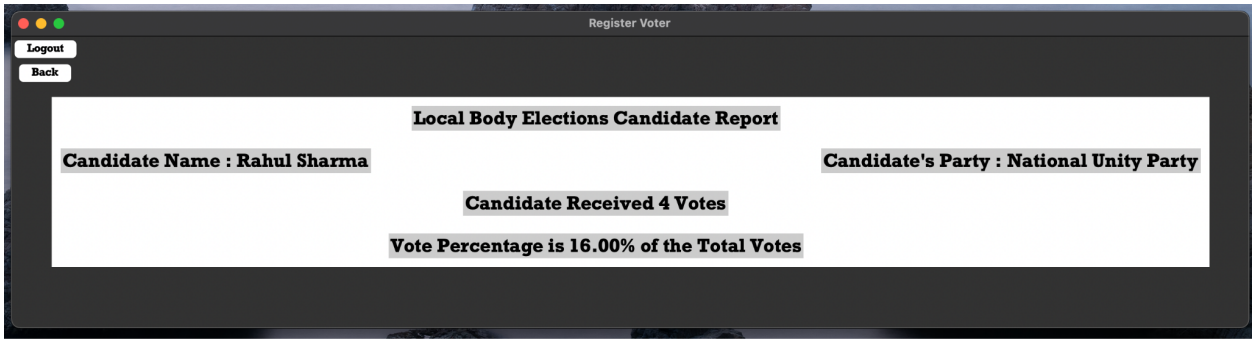
→ Query to Retrieve the Percentage of Votes Received by a Candidate in a Given Election

```
-- Query To Calculate Percentage of Votes Received By a Candidate in an Election
SELECT COUNT(*)*100/(SELECT COUNT(*) FROM VOTE WHERE Election_ID = 1)
FROM VOTE WHERE Candidate_ID = 1 AND Election_ID = 1;
```

This Query is used to give us the Percentage of Votes For a **Candidate** in a given **Election**. It takes the following parameters:

- **Candidate_ID**: The Candidate_ID of the Candidate the Vote is Cast For (INT).
- **Election_ID**: The Election_ID of the Election the Vote is Cast In (INT).

It gives us the Percentage of Votes For a Candidate. In this example, the Candidate Rahul Sharma got **16% of the Total Votes** in the Local Body Elections.



→ Query to Get the Winner of an Election

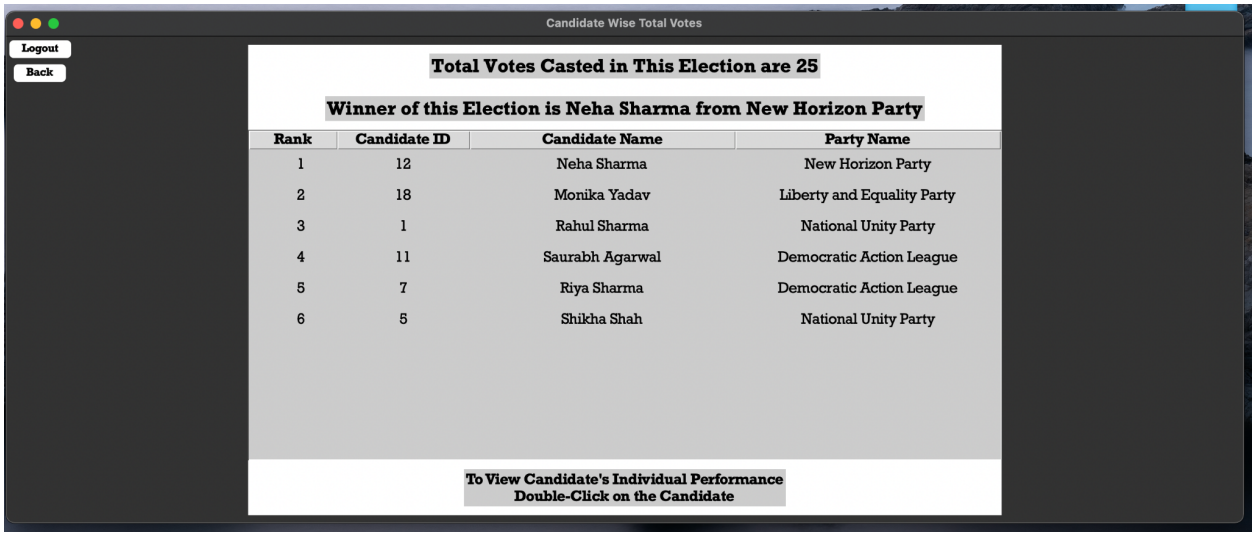
```
-- Query To Determine Winner of an Election
SELECT Name,Party_Name  FROM Candidate NATURAL JOIN (SELECT Candidate_ID,COUNT(Candidate_ID)
AS VOTE FROM VOTE WHERE ELECTION_ID = 1 GROUP By Candidate_ID ORDER BY VOTE DESC)
as T ORDER BY VOTE DESC LIMIT 1;
```

This Query is used to tell us the Winner of a given **Election**. It takes the following parameters:

- **Election_ID**: The Election_ID of the Election the Vote is Cast In (INT).

This Query cannot determine the Winners of Ongoing and Upcoming Elections, and the Front-End has handled all exceptions.

It gives us the Winner of the given Election. In this example, **Candidate Neha Sharma** won the Local Body Elections.



→ Query to Retrieve the Rank of Each Candidate Based on the Number of Votes Received

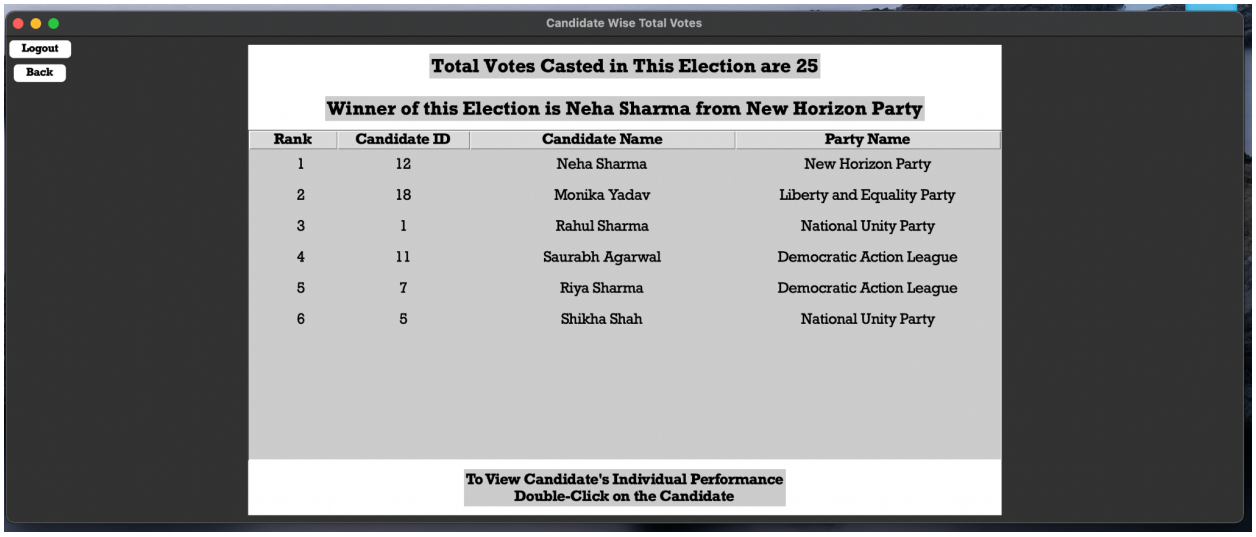
```
-- Query To Determine Rank List of Candidates for an Election
SELECT Candidate_ID,Name,Party_Name  FROM Candidate NATURAL JOIN (SELECT Candidate_ID,COUNT(Candidate_ID)
AS VOTE FROM VOTE WHERE ELECTION_ID = 1 GROUP By Candidate_ID ORDER BY VOTE DESC) as T ORDER BY VOTE DESC;
```

This Query is used to tell us the Rank List of a given **Election Result**. It takes the following parameters:

- **Election_ID**: The Election_ID of the Election the Vote is Cast In (INT).

This Query cannot determine this for Ongoing and Upcoming Elections, and the Front-End has handled all exceptions.

It gives us the Rank List of the Candidates. In this example, you can see the Ranks using the Left-Most Column.



→ Query to Retrieve the Total Number of Votes Cast for an Election.

```sql
-- Query To Get Total Votes Made in an Election
SELECT COUNT(*) FROM VOTE WHERE Election_ID = 1;
```

This Query is used to tell us the total number of votes cast in an **Election**. It takes the following parameters:

- **Election_ID**: The Election_ID of the Election the Vote is Cast In (INT).

This Query cannot determine this for Ongoing and Upcoming Elections, and the Front-End has handled all exceptions.

It gives us the Total Number of Votes Cast in an Election. In this example, you can see that **Total Votes Casted is 25**.

→ Query To Update a Voter's Information

```sql
-- Query To Update a User's Profile
UPDATE Voter SET Password = 'password1', Name = 'Amit Singh Khurana', Email = 'amit.singh@example.com',
Address = '34/25, Shyam Nagar, 201001', Phone_Number = '9767901970', DateOfBirth = '1990-01-01'
WHERE Voter_ID = 1;
```

This Query is used to Update a Voter's information in the **Voter** Table. It takes the following parameters:

- **Password**: The Password of the Voter (VARCHAR(50)).

- **Name**: The Name of the Voter (VARCHAR(50)).
- **Email**: The Email Address of the Voter (VARCHAR(50)).
- **Address**: The Home Address of the Voter (VARCHAR(100)).
- **Phone_Number**: The Phone Number of the Voter (CHAR(10)).
- **DateOfBirth**: The Date of Birth of the Voter (DATE).

It updates the data accordingly. Integration is such to catch all cases of Information in Invalid Formats and Showcase the Errors to the User to fix. Highly Controlled Database Access. Great help in validating Queries with a DATE type parameter.

→ **Query To Delete A Voter's Information**

```
-- Query To Delete a Specific Voter
DELETE FROM VOTER where VOTER_ID = 16;
```

This Query is used to Delete a Voter's information from the **Voter** Table. It takes the following parameters:

- **Voter_ID**: The **Voter_ID** of the Voter whose Information is to be deleted (INT).

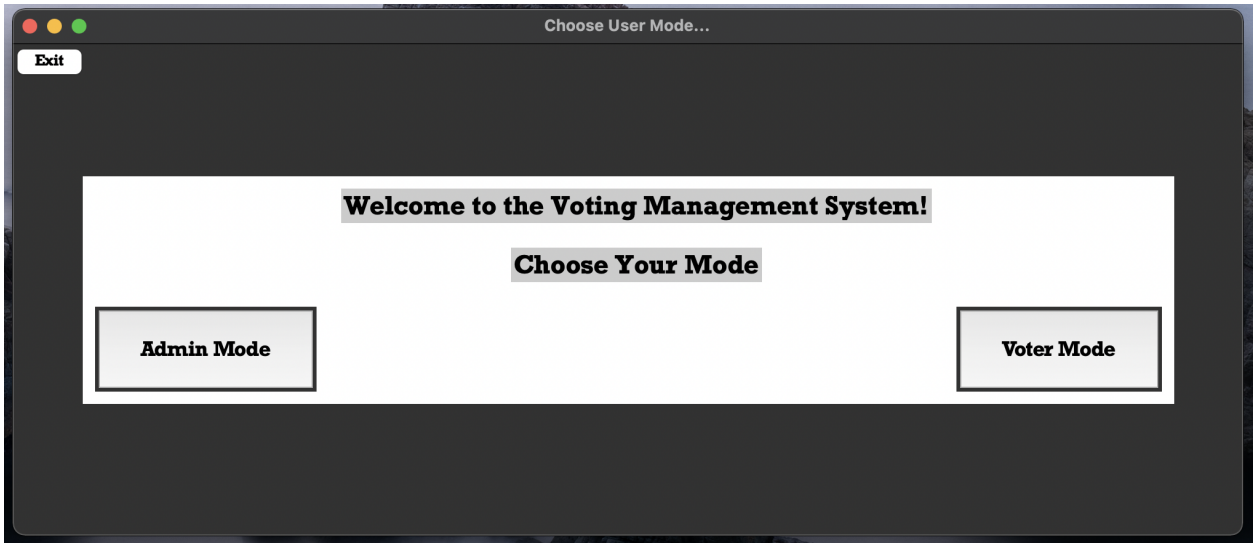
It updates the data accordingly.

FrontEnd and BackEnd

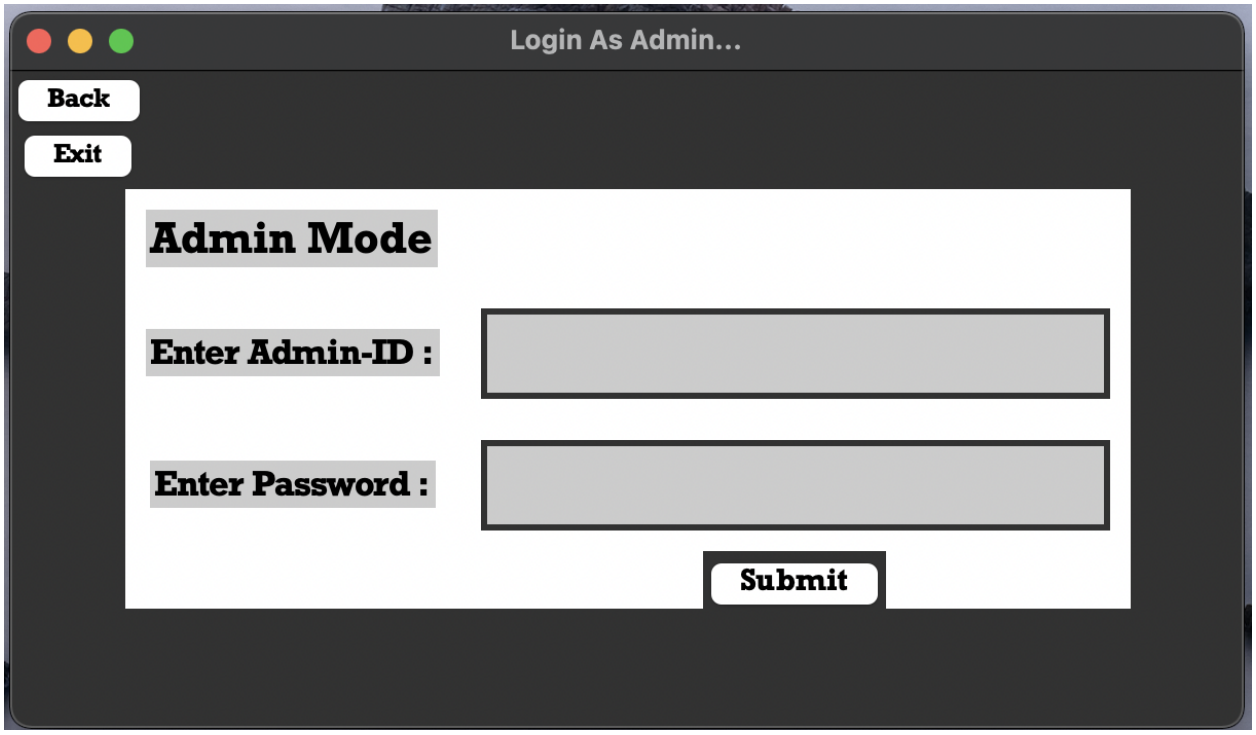
- TechStack Used for this Project MySQL, MySQL.Connector, Python, Tkinter.
- Installment of these would be necessary to run the application.
- Backend is written fully in Python+MySQL, with extensive use of MySQL.Connector resources.
- For further information, refer to the file "Queries.py", "Queries2.sql" and "Procedures.sql" which has SQL implementation of the functionalities used in the application.
- You can view the SQL Statements for Each Function On your Terminal

Front-End Documentation

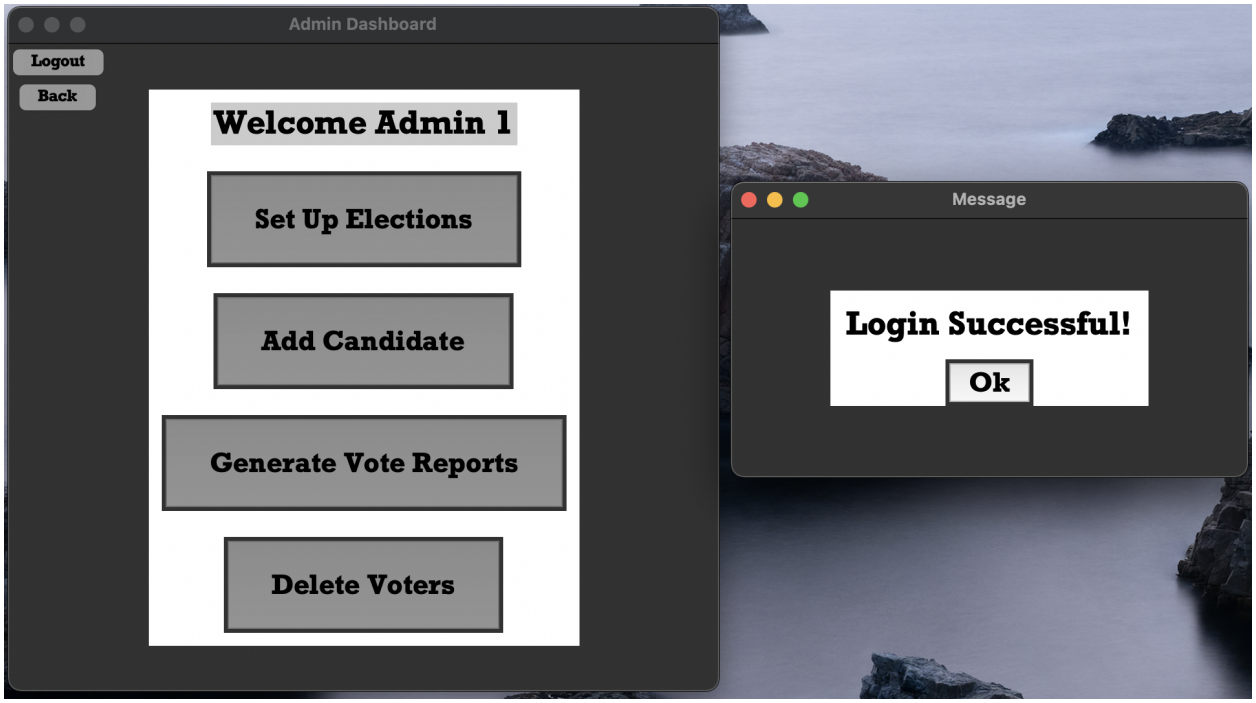
→ User Mode Page



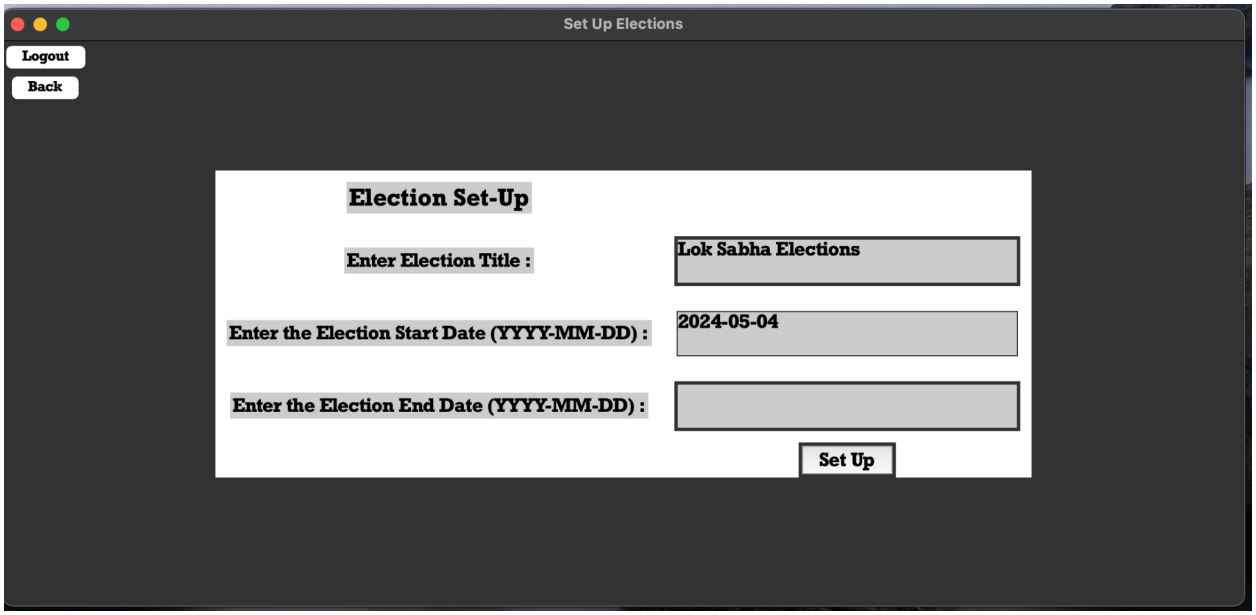
→ Admin Login Page



→ **Admin DashBoard**

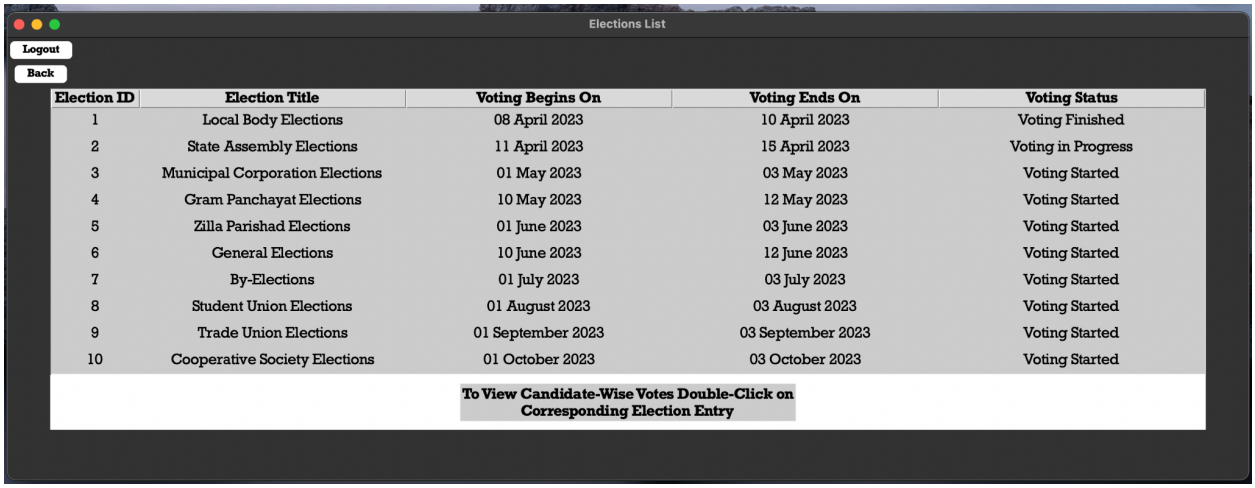


→ Election Set Up Page

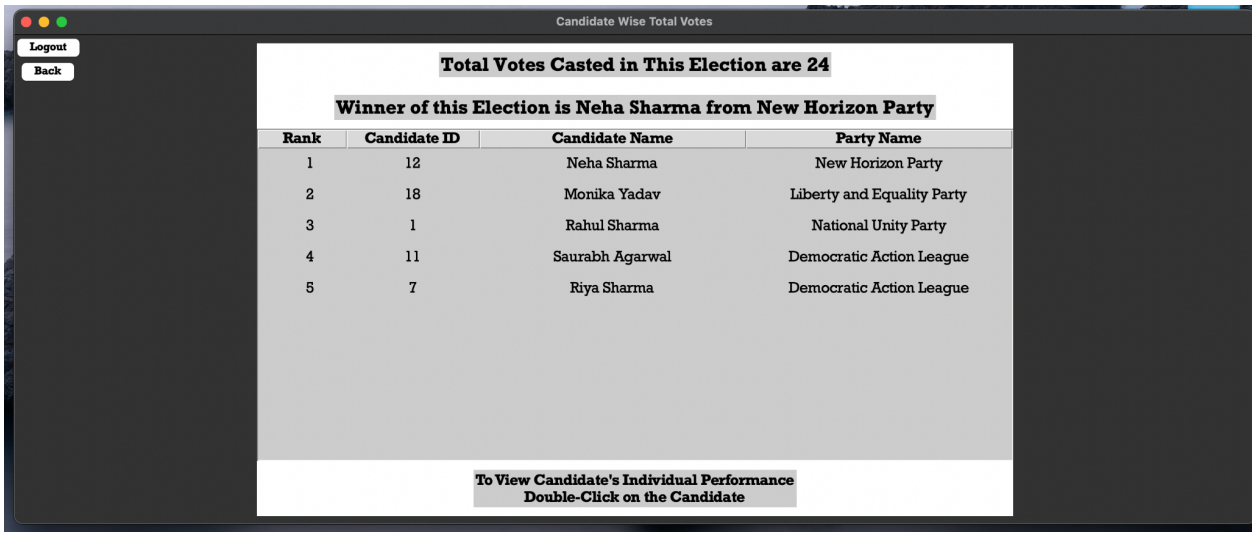


→ Add Candidates Page

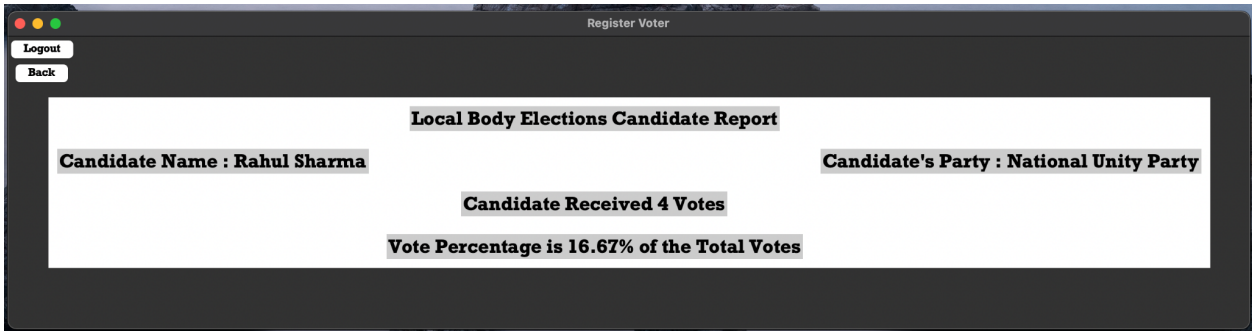
→ Generate Vote Reports, List of Elections



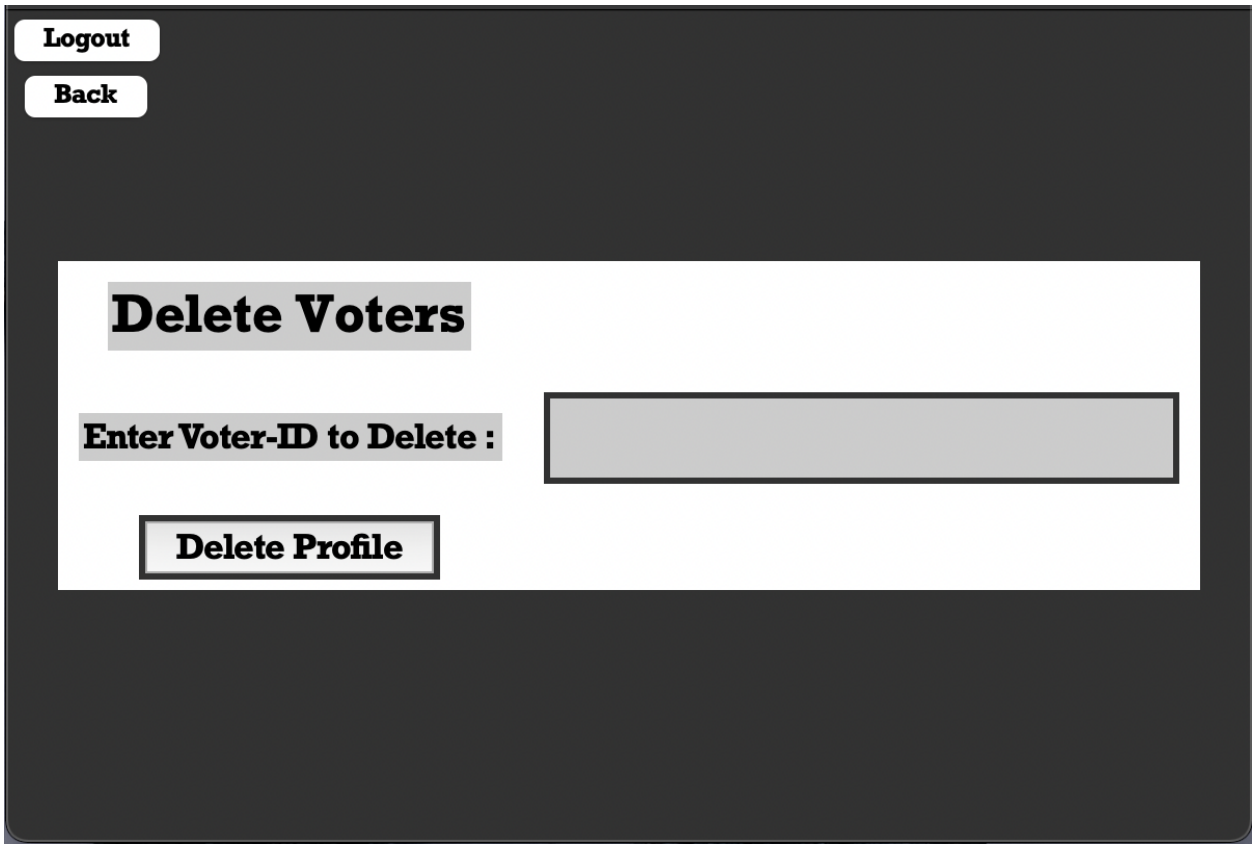
→ Candidates Rank List



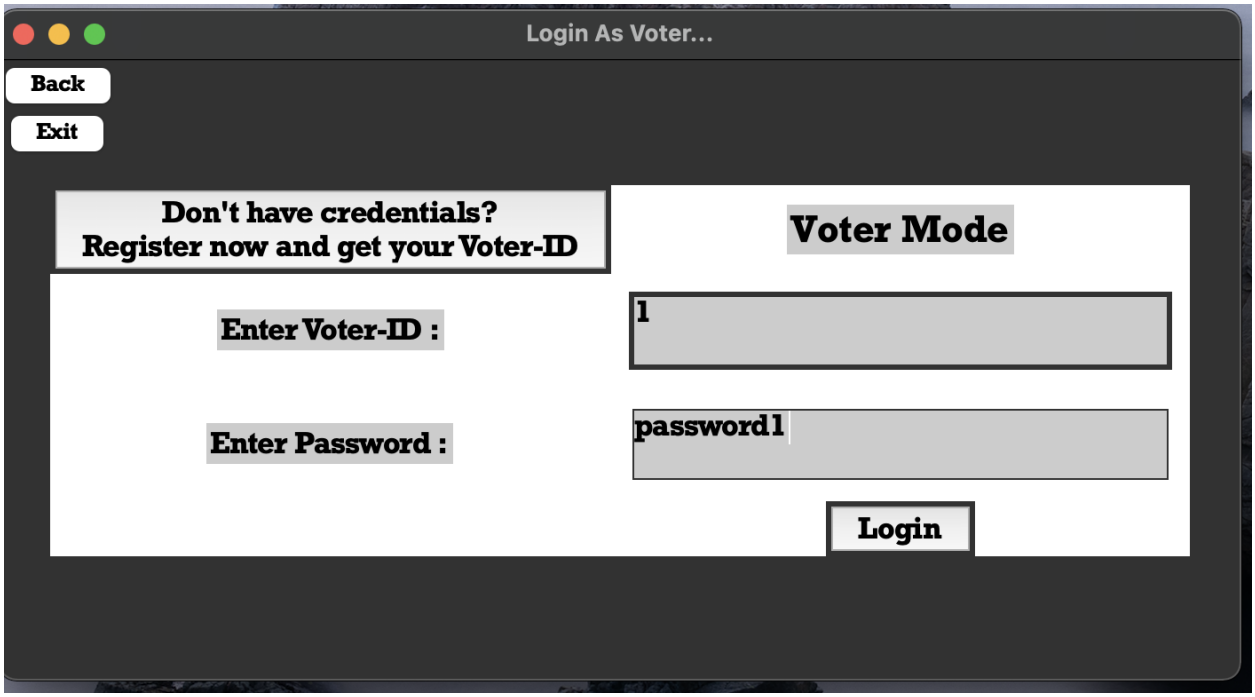
→ Candidate Wise Vote Report



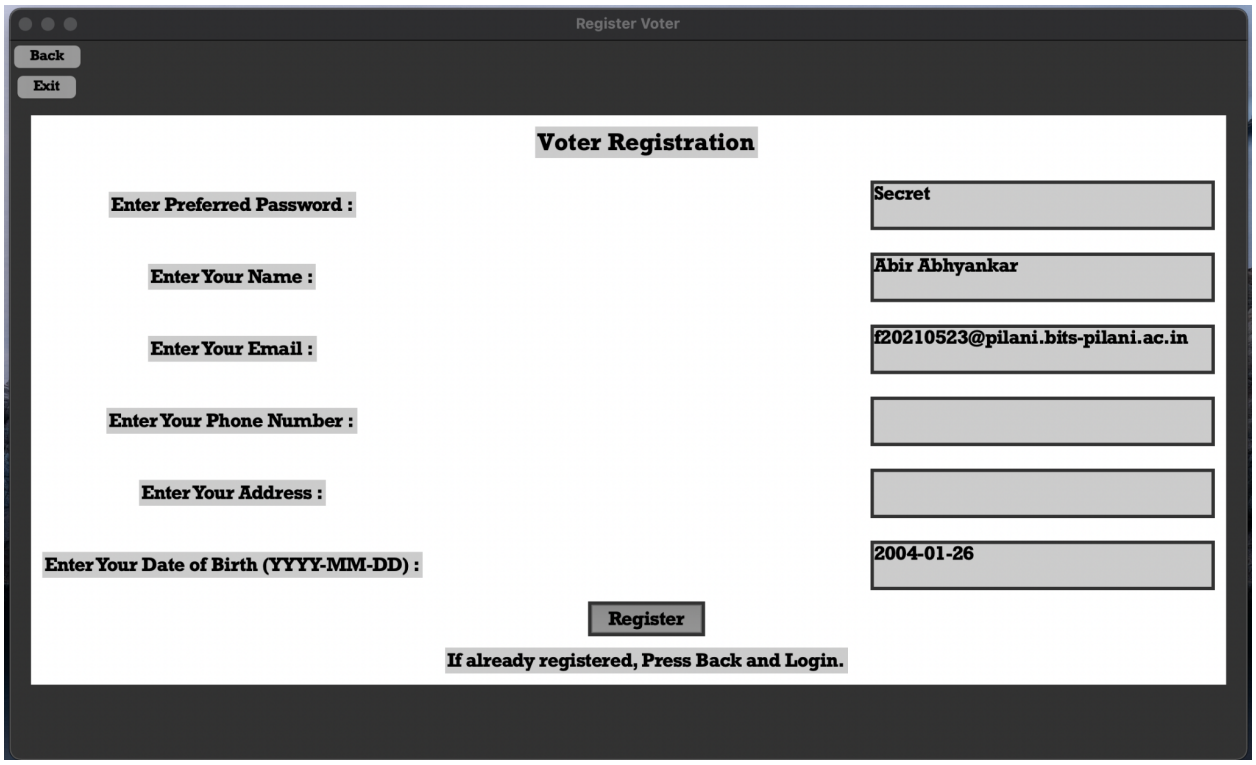
→ Delete Voter Profiles



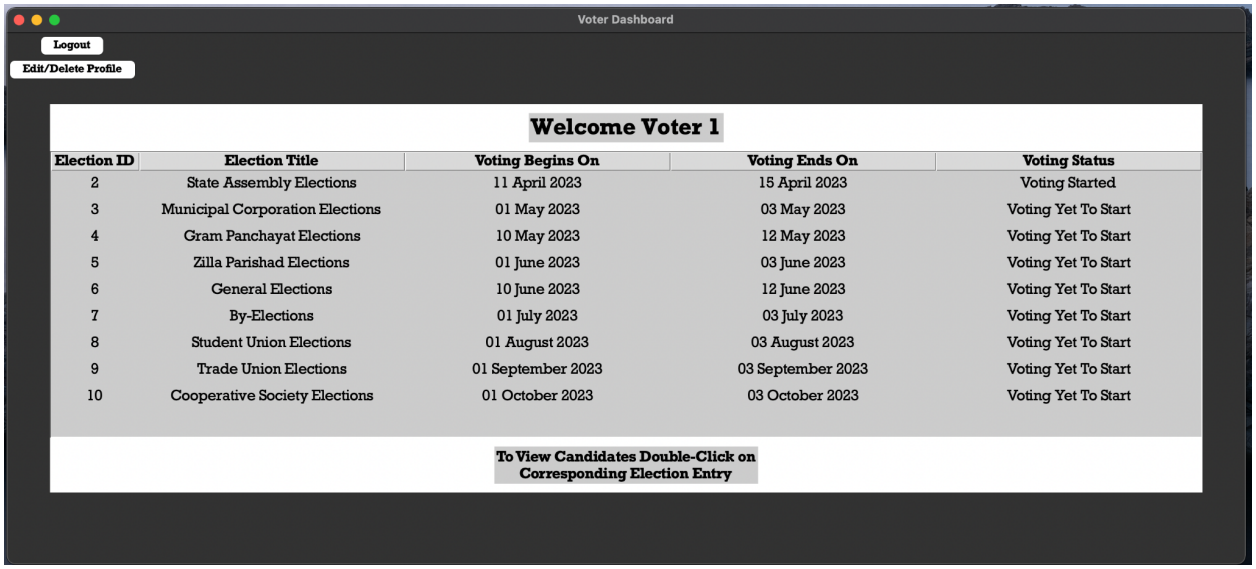
→ Voter Login



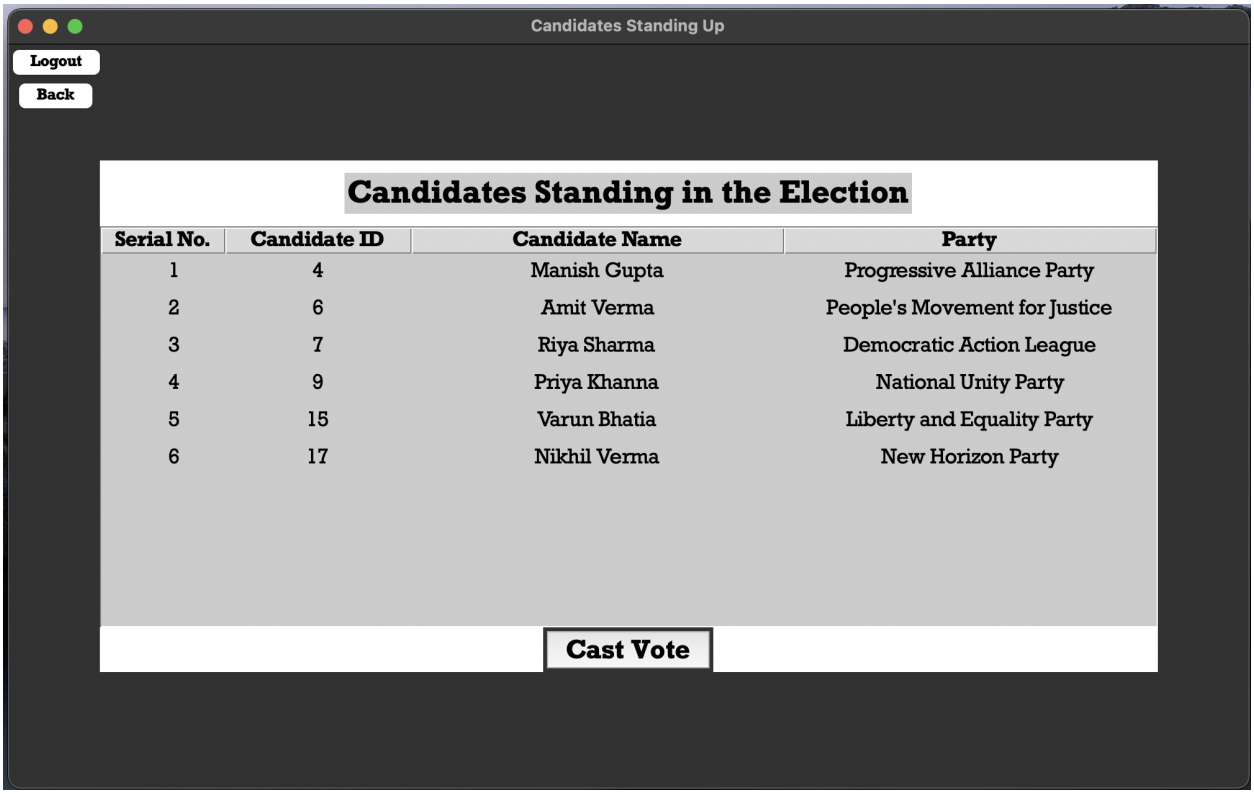
→ Voter Registration



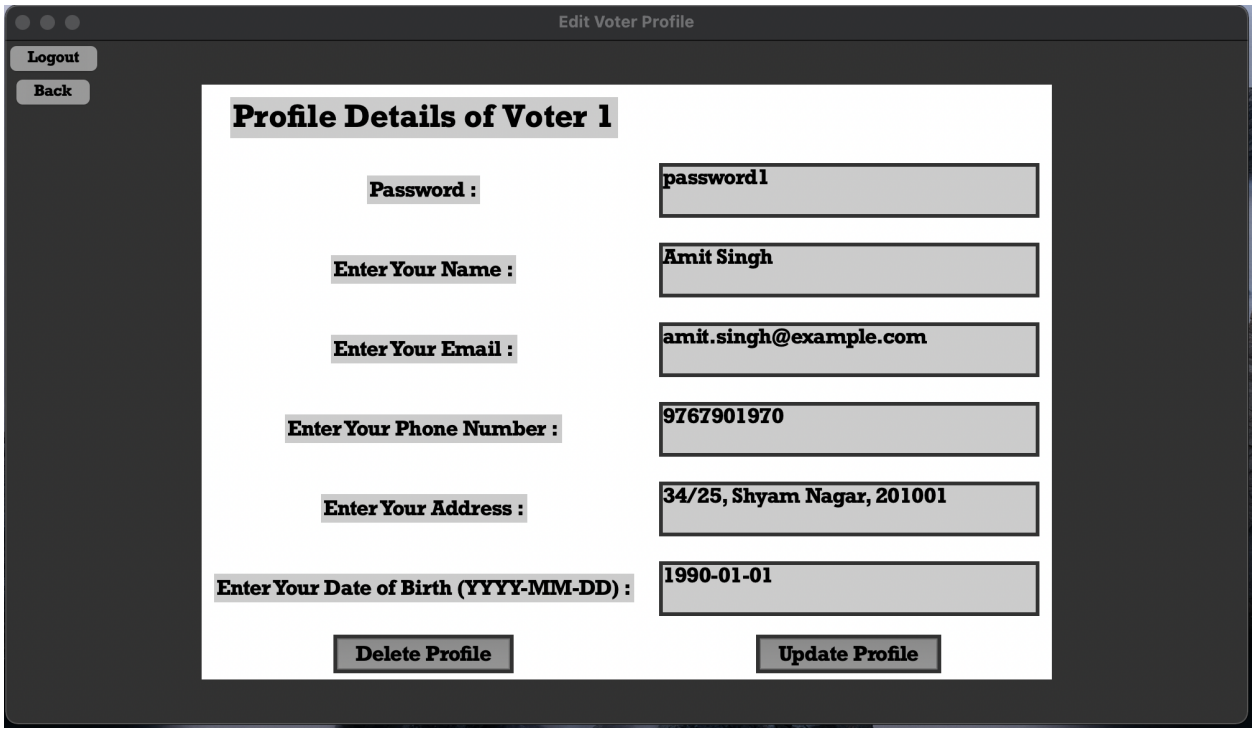
→ Voter Dashboard



→ Candidate List Standing for an Election



→ **Edit Profile Page**



ABIR ABHYANKAR
ID - 2021A7PS0523P